#include<stdio.h>

#include<stdlib.h>

const int N = 10;

struct stack {

int arr[N];

int size;

};

void stackInit(struct stack& koko) {

koko.size = 0;

}

int empty(struct stack& koko) {

if (koko.size == 0)

return 1;

else

return 0;

}

void push(struct stack& koko, int x) {

if (koko.size == N) {

printf("stack full!");

}

if (koko.size < N && koko.size > -1) {

koko.arr[koko.size] = x;

koko.size++;

}

}

int stacktop(struct stack& koko) {

if (empty(koko) == 0)

return koko.arr[koko.size - 1];

}

int pop(struct stack& koko) {

if (empty(koko) == 0) {

int x = koko.arr[koko.size - 1];

koko.size = koko.size - 1;

return x;

}

}

void print(struct stack& koko) {

for (int i = 0; i < koko.size; i++) {

printf("%d,", koko.arr[i]);

}

}

void Delete(struct stack& koko, int x) {

struct stack s2;

stackInit(s2);

int counter = 0;

if (empty(koko) == 0)

{

while (stacktop(koko) != x) {

push(s2, pop(koko));

counter++;

}

pop(koko);

while (counter > 0) {

push(koko, pop(s2));

counter--;

}

}

else

printf("the stack is empty!");

}

int search(struct stack& koko, int x) {

struct stack s2;

stackInit(s2);

int counter = 0;

if (empty(koko) == 0) {

while (stacktop(koko) != x) {

push(s2, pop(koko));

counter++;

}

}

int j = counter;

while (counter > 0) {

push(koko, pop(s2));

counter--;

}

return j + 1;

}

struct QueueStack {

struct stack s1;

struct stack s2;

};

void QueueInit(struct QueueStack& Qs) {

stackInit(Qs.s1);

}

int isEmpty(struct QueueStack& Qs) {

int x = empty(Qs.s1);

return x;

}

int isFull(struct QueueStack& Qs) {

if (Qs.s1.size == N - 1)

return 1;

else

return 0;

}

void enqueue(struct QueueStack& Qs, int number) {

push(Qs.s1, number);

}

int dequeue(struct QueueStack& Qs) {

stackInit(Qs.s2);

int counter = 0;

int x = 0;

if (isEmpty(Qs) == 0)

{

while (Qs.s1.size > 0)

{

push(Qs.s2, pop(Qs.s1));

}

x = pop(Qs.s2);

while (Qs.s2.size > 0)

{

push(Qs.s1, pop(Qs.s2));

}

return x;

}

else

printf("the Queue is Empty!");

}